Technical Document 1454
January 1989

Temporal Pattern Recognition

A Network Architecture for Multisensor Fusion

C. E. Priebe
D. Marchette

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This task was performed for the Chief of Naval Operations, Washington, DC 20350, under program element 0602232N, project CDB2, agency accession DN306 242. This paper was first presented at the Cambridge Symposium on Advances in Intelligent Robotic Systems, Cambridge, MA, November 1988.

Released by
V. J. Monteleon, Head
Architecture and Applied
Research Branch

Under authority of
J. A. Salzmann, Jr., Head
Information Systems
Division

LH

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC Technical Document 1454	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center 6b. OFFICE SYMBOL (if applicable) Code 421	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Chief of Naval Operations 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Washington, DC 20350	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 0602232N PROJECT NO.: CDB2 TASK NO.: AGENCY ACCESSION NO.: DN306 242

11. TITLE (include Security Classification)
TEMPORAL PATTERN RECOGNITION
A Network Architecture for Multisensor Fusion

12. PERSONAL AUTHOR(S)
C.E. Priebe and D. Marchette

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM TO	January 1989	14

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
			network architecture multisensor fusion temporal pattern recognition

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper proposes a self-organizing network architecture for the learning and recognition of temporal patterns. This architecture is used to perform multisensor fusion and scene analysis for ROBART II, a second-generation autonomous sentry robot.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL C.E. Priebe	22b. TELEPHONE (include Area Code) (619) 553-4048	22c. OFFICE SYMBOL Code 421

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

Temporal Pattern Recognition:
A Network Architecture for Multi-Sensor Fusion

Carey E. Priebe
and
David Marchette

Architecture and Applied Research Branch
Code 421
Naval Ocean Systems Center
San Diego, CA 92152
Phone: (619) 553-4048 / 4049

ABSTRACT

A self-organizing network architecture for the learning and recognition of temporal patterns is proposed. This multi-layered architecture has as its focal point a layer of multi-dimensional Gaussian classification nodes, and the learning scheme employed is based on standard statistical moving mean and moving covariance calculations. The nodes are implemented in the network architecture by using a Gaussian, rather than sigmoidal, transfer function acting on the input from numerous connections. Each connection is analogous to a separate dimension for the Gaussian function. The learning scheme is a one-pass method, eliminating the need for repetitive presentation of the teaching stimuli. The Gaussian classes developed are representative of the statistics of the teaching data and act as templates in classifying novel inputs. The input layer employs a time-based decay to develop a time-ordered representation of the input stimuli.

This temporal pattern recognition architecture is used to perform multi-sensor fusion and scene analysis for ROBART II, a second-generation autonomous sentry robot employing heterogeneous and homogeneous binary (on / off) sensors. The system receives sensor packets from ROBART indicating which sensors are active. The packets from various sensors are integrated in the input layer. As time progresses these sensor outputs become ordered, allowing the system to recognize activities which are dependent, not only on the individual events which make up the activity, but also on the order in which these events occur and their relative spacing throughout time.

Each Gaussian classification node, representing a learned activity as an ordered sequence of sensor outputs, calculates its activation value independently, based on the activity in the input layer. These Gaussian activation values are then used to determine which, if any, of the learned sequences are present and with what confidence. The classification system is capable of recognizing activities despite missing, extraneous or slightly out-of-order inputs. An important predictive quality is also present. This system can predict that an activity may be about to occur prior to receiving confirmation that all component events have occurred.

Overall, the temporal pattern recognition system described allows the robot to go beyond the alert / no alert stage based on a simple weighted count of the sensors firing. ROBART is now able to determine which activities are occurring, enabling it to intelligently act on this information.

CONTENTS

Temporal Pattern Recognition:
A Network Architecture for Multi-Sensor Fusion

C. E. Priebe
and
D. J. Marchette

Architecture and Applied Research Branch
Code 421
Naval Ocean Systems Center
San Diego, CA 92152
Phone: (619) 553-4048 / 4049

I. Introduction

The ability to understand one's environment, the goal of a multi-sensor fusion system, is not governed solely by static pattern recognition. The order in which sensor firings occur can be as important as the events themselves, and an intelligent robotic system must be able to detect and understand this ordering. Thus the dimension of time allows access to a wealth of information about the current environment, past events, and expectations about the future. Thus the ability to incorporate time into information processing is essential for multi-sensor fusion, and is necessary for robotic tasks such as the recognition of sequences of events, understanding cause and effect, making predictions and planning.

Multi-sensor fusion involves taking data from a variety of sensors and using this data to construct an overall picture of the environment. In particular, we will be concerned with the problem of determining the activities present from the pattern of sensor firings of a robot. For instance, the robot should be able to distinguish between an intruder entering the room and a thunder and lightning storm. In both cases there may be noises and flashing lights, but the pattern of sensor firings will be different in the two cases. For the purposes of this paper, all sensors will be binary. No information is conveyed by the sensor firing other than the fact that the sensor has detected its activator. Thus a light sensor will fire if the amount of light reaching it is above a threshold.

The primary requirment of a multi-sensor fusion system is the ability to recognize sequences. This ability is essential for many tasks, notably those involved with audition and vision, as well as activity recognition. A sequence may consist of a stream of phonemes, typed letters or sensor outputs. Once the initial preprocessing has been done and the individual members of the sequence have been recognized, the task is to determine which of the known sequences are represented by the input. Since the answer may depend on the context in which the input has been received, the system must return all the sequences that the input might represent with a confidence rating indicating the quality of match between the input and the known sequences.

The architecture described is a type of adaptive network system, also called "neural network" by some. The system consists of a large number of independent processing elements and connections between them through which information is passed. This gives the system an inherently parallel structure which allows real-time processing and a measure of fault tolerance due to redundancy.

The network used for robotic activity recognition runs on a 80386-based computer. It consists of three layers of processing elements. Data comes from the robot in sensor packets indicating which sensors fired. The input into the first layer is temporally ordered using a decay scheme. A Gaussian classification layer recognizes sequences of events, and the output layer sums the output of one or more Gaussians to produce a more general representation of the sequences.

The application described here uses ROBART II, a mobile sentry robot resident at Naval Ocean Systems Center, as the testbed. ROBART II, described in [2], has multiple homogeneous and heterogeneous binary sensors, and the task is to accept input from these diverse sensors, fuse this incoming data, and develop an understanding of the activities taking place in the environment. To do this, the Gaussian network must be able to develop a representation of actvities of interest that incorporates time as well as sensor type.

II. Gaussian Classification

Given a stream of sensor reports, a class must be assigned to the data indicating the activity that is occuring. This activity is dependent not only on which sensors fired, but on the order in which the sensors fire, and even the timing between sensor firings. In addition to the assignment of a class, it is desirable to give a probability or likelihood that the assignment is correct. In general, some measure of likelihood is given for all classes known to the system. Since more than one activity can occur at any time, the elements of the vector need not sum to one. This vector of likelihood measures will be referred to as the classification vector. It is this vector which is returned by the system.

The approach to classification taken in this paper derives from the field of density estimation. The idea is to approximate the probability density functions of the classes and use them to assign a classification vector to novel input. It is important to stress that the classification vector is not a vector of probabilities, but rather probability densities, and thus need not even be bounded by one (though the values are constrained to be non-negative).

The Gaussian or normal distribution (1) is used extensively in density estimation [6]. In parametric density estimation, the distribution is assumed to be a known one, in this case a Gaussian, and the parameters of the distribution are estimated from the data. A more general approach is to approximate the distribution as a sum of Gaussians. This allows the modelling of a wide range of distributions. The kernel estimator [6] is the most commonly used method of this type. The architecture described herein is an adaptation of the kernel estimator.

The multidimensional Gaussian is the familiar multivariate normal distribution:

$$G(\underline{x}) = \frac{\exp\left(-0.5 \left[(\underline{x} - \underline{\mu})^t \cdot \Sigma^{-1} \cdot (\underline{x} - \underline{\mu}) \right] \right)}{(2\pi)^{(d/2)} * |\Sigma|^{(1/2)}} \qquad (1)$$

This uses the Mahalanobis distance (the quantity in square brackets in (1)) between its input and the stored mean to compute a Gaussian distance. Here, $\underline{x}$ is the a real valued vector of dimension d, $\underline{\mu}$ is the mean of the Gaussian, and Σ is the covariance matrix.

A useful interpretation of the Gaussians in this scheme is as basic feature detectors. Each Gaussian can be thought of as keying on a particular feature within the data, and the number of Gaussians used for the data determines the number of features used to distinguish the data. In this interpretation, the means of the Gaussians become the feature, and each Gaussian returns a distance measure between its feature and the input data item. It is instructive to think of the Gaussians in this manner, particularly as it gives them an autonomous flavor. Each Gaussian is conceived of as performing a test of its input against a stored feature, making this computation independantly of the others, and returning a measure of the distance between the input and its feature. It is this independence that is critical to implementation as a network architecture, and leads to the architecture that will be described.

III. Network Architecture

The actual computational architecture proposed is designed explicitly for massively parallel processing. A preliminary description of this work can be found in [4]. We describe the system as a collection of nodes, organized in layers, and connections between these nodes (Figure 1).

The first layer is the input layer, which acts as the bridge between the network and the outside world. The sensor outputs are the input to this layer, and its output is a representation of the order of firing for each sensor. This temporal representation is implemented by decaying the output of the nodes, so that the most recent input has the highest output value.

The middle, or hidden, layer consists of Gaussian classifiers. These are nodes whose transfer function is a Gaussian, and they return the Gaussian distance between their stored pattern and the input. These Gaussians have been modified to allowfor the comparison of nodes with different input dimension.

The final layer computes a weighted sum of the outputs of the Gaussian nodes of each class. Each Gaussian is weighted by the percentage of points from its class that have been used to update that node.

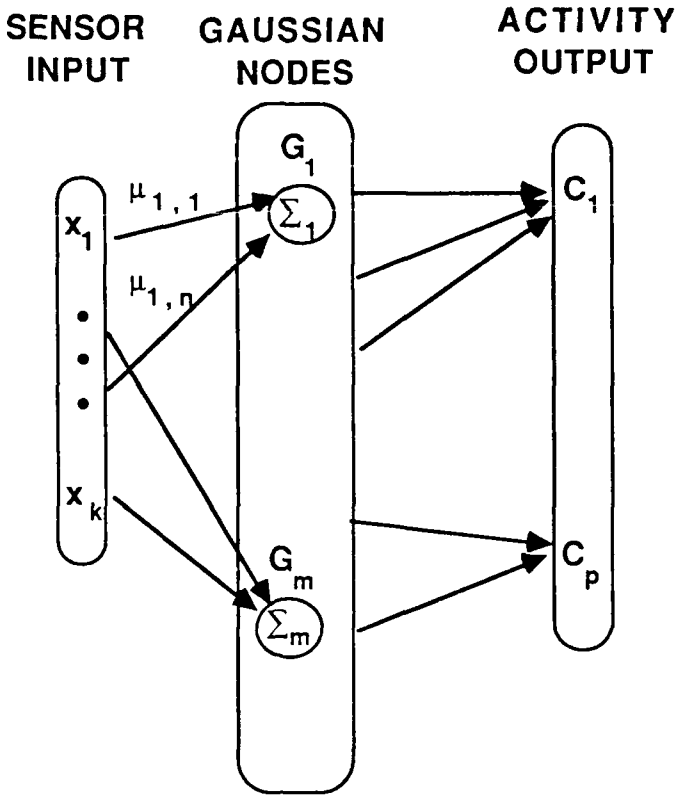


Figure 1
Layered Gaussian classification architecture.

Individual nodes

The input nodes perform the temporal ordering of the sensor firings. This allows the network to learn and recognize sequences of events. The temporal ordering scheme used is straightforward [3]. When input stimuli are received into the input field from the various sensors, the activation values in the field experience a decay factor that acts as an ordering function. By using this decay (as seen in Figure 2) the input field can develop a representation in which the order the stimuli were received is preserved. This decay is implemented as follows:

$$a_j(t+1) = a_j(t) - \Gamma * a_j(t) \qquad (2)$$

where $a_j(t)$ is the activation value at time t for input node j, and Γ is the decay rate. Γ is chosen large enough to sufficiently seperate the inputs for the Gaussian classification. This ordering can then be used to determine the distance of the current input field from the learned patterns already represented on the Gaussian level.

It is clear that the input field must contain more than a single node for each distinct stimuli. For example, if the system is categorizing sequences of letters, there must be more than one input node corresponding to the letter "A". Were this not the case, the system could not handle sequences with more than one "A" in them, as the second "A" would overwrite the first. This can be mitigated to some extent by feeding back the previous outputs for the hidden layer as input, allowing a kind of "memory" in the system (3).

Each node in the hidden layer is a Gaussian classification node, so named because the fundemental property of a network node, the transfer function, is a Gaussian function (1). As such, each node requires two defining characteristics: a covariance matrix Σ and a mean $\underline{\mu}$. Figure 1 indicates the implementation of these defining characteristics. The covariance matrix for a node G_j is contained at the node, while the mean is defined as the n-tuple $<\mu_{1,j}, \ldots, \mu_{n,j}>$ of afferent connection weights coming into G_j from the previous layer.

After input of: A B C D E
Decay at Input level: $\Gamma = 0.15$
Input layer is:

A = .5220
B = .6141
C = .7225
D = .8500
E = 1.000

Figure 2
Example of decay in input layer.

In practice, it is sometimes necessary to deviate from the strict Gaussian activation function and use a pseudo-Gaussian. By this we mean a transfer function retaining the basic Gaussian shape but not necessarily retaining other properties of the true Gaussian. For robotic activity recognition the output of two or more nodes must be compared despite their inequality in dimensionality. This inequality of dimension comes about for two reasons: different activities can have different numbers of events, and the system must work with incomplete data. That is, it must give an output even if all the events necessary for an activity have not yet been received. This can cause problems in the standard Gaussian scheme, since if two Gaussians have different dimensions, the smaller dimensional Gaussian will have a larger output for a point at its mean than the larger dimensional Gaussian will for a point at its mean. The general form for the pseudo-Gaussian activation function is described in equation (3).

$$G_j(t+1) = \Delta G_j(t) + B_j * D_j * e^{-\eta E_j} \qquad (3)$$

This equation gives the activation of of Gaussian node G_j at time t+1 in terms of the node's previous activation, $G_j(t)$, and an exponential function of E_j, the Mahalanobis distance between the input stimulus $\underline{x}$ and the mean $\underline{\mu}$ of Gaussian G_j defined in equation (4).

$$E_j = (\underline{x} - \underline{\mu})^t \cdot \Sigma^{-1} \cdot (\underline{x} - \underline{\mu}) \qquad (4)$$

Δ is a short-term memory constant which allows an added dimension of history - the node's previous activation - to be incorporated into the activation calculation. η is a Gaussian parameter that alters the default standard deviation -- the width of the bell-shaped curve -- of the Gaussian function. B_j is a statistical parameter based on the number of patterns seen. This parameter can give an a priori estimate of the probability that the current stimulus belongs to the category represented by node G_j. In it simplest form, B_j equates to the ratio of the total number of patterns seen to the number of patterns that have been categorized as belonging to the category indicated by node G_j.

D_j is a function of the dimension of the current stimulus and the (static) dimension of Gaussian G_j. Since not all Gaussian nodes will have the same dimensionality, and because we require the system to begin to recognize a pattern prior to receiving the entire pattern, it is necessary to include a weighting factor into the activation function of the Gaussian nodes. Because this dimensionality parameter is a function of the current input stimulus' dimensionality, the system is able to determine that it does not have a perfect match at node G_j even in the case where the inputs match perfectly the first few elements of the sequence stored at node G_j.

The hidden layer is essentially a feature detection layer consisting of Gaussian classification nodes. It is in these feature detection Gaussians nodes, and in their afferent connections, that the learning takes place. The Gaussian layer (G1) is made up of nodes with afferent connections coming from the input layer. An initial classification, or partitioning of the input space, is accomplished at G1.

The output layer is not actually a feature detection layer. At this layer the network performs a "combination of Gaussians" (Figure 1). This technique, taken from kernel estimation theory [6], allows the network as a whole enormous flexibility. The nodes in the output level correspond one-to-one to the classes being

discriminated. The non-zero afferent connections into an output node C_j from layer G1 indicate the features, or preliminary Gaussian classifications, that are to be combined to make up the ultimate distribution for class j. While each of the G1 nodes that are summed to create C_j are themselves Gaussians, the final distribution need not be constrained to a Gaussian. A wide variety of distributions can be approximated by a mixture of Gaussians. The network has the flexibility to model a particular class based on the statistics of the input that the system has seen, rather than being forced to fit the data to a Gaussian.

Upon receiving an input stimulus, the information propagates through the network, driven by local nodal calculations, ultimately determining the activation values of the output nodes, and hence the individual class values. These outputs are then interpreted as the relative likelihoods that the current input belongs to the individual classes. Figure 3 gives some examples of interpretations of network outputs.

$C_1 = 0.17$	$C_1 = 0.23$	$C_1 = 0.19$
$C_2 = 0.97$	$C_2 = 0.11$	$C_2 = 0.83$
$C_3 = 0.24$	$C_3 = 0.13$	$C_3 = 0.81$
(a)	(b)	(c)

Figure 3
Example outputs from a 3 class system.

The network's output values yield a .ual interpretation. First, the relative values are important. The output node with the highest activation value corresponds to the class to which the network believes the input belongs. In addition to this relative interpretation, there is an absolute one. For Figure 3a, the system is indicating unambiguously that it has detected class 2. In Figure 3b node C_1 has the highest value and class 1 is therefore the network's best guess as to the classification of the input. However, the low value of this winning node indicates a lack of confidence in this classification. It is likely that the input stimulus used to evoke this output belongs to an entirely new class -- a class for which the network has not developed a representation. Similarly, in Figure 3c, two unique classes are indicated with high likelihood. This implies that, although one of the nodes (C_3) has the highest value, both classes 2 and 3 are potentially correct classifications for the input. In other words, the sequences which represent classes 2 and 3 are detected in the input.

It is now possible to see that each node is independent of the other nodes on the same layer. This implies that all the nodes or a given layer, say G1, can process in parallel. Since a layered system such as that depicted in Figure 1 can be thought of as a pipeline processing architecture, the entire system, when realized in parallel hardware, can accept inputs with a relatively fine discrete time step. The delay between successive inputs need be no longer than the time it takes to calculate a multi-dimensional Gaussian function. The slowest individual node in the system defines the time step, and that node need perform only a single, simple calculation.

Learning techniques

While the information flow through the network defines the classification architecture, it is still necessary to describe the rules for network adaptation. It is unacceptable to require the system be handcrafted for a particular problem. There must be a method allowing the network to dynamically adapt to incoming stimuli, developing an internal representation for the statistics of the stimuli. This internal representation is necessarily created from layers of Gaussian feature detectors and the eventual summation of Gaussians, as described above.

To develop the internal representation of the data, the Gaussians must be chosen with means and covariance matrices appropriate to the data. The formulas for iteratively computing these parameters are as follows. For clarity, we will focus on a single Gaussian node, and assume that the decision to update the node has been made. Consider the mean $\mu(n)$, constructed from the first n points. Then $\mu(n+1)$ is formed by adding to its components

$$\Delta\mu_i = \frac{1}{n+1} (X_{(n+1)i} - \mu_i(n)) \qquad (5)$$

where $X_{(n+1)i}$ is the i^{th} component of the $n+1^{st}$ data point. Equation (5) gives the amount that the mean should be changed to include a new data point. A similar formula can be constructed from the sample covariance. Let

$$S_{ij} = \Sigma(X_{ki} - \mu_i)(X_{kj} - \mu_j) \tag{6}$$

where X_{kl} is the l^{th} component of the k^{th} data point. Then (see [1])

$$\Delta S_{ij} = \frac{n}{n+1} (X_{(n+1)i} - \mu_i(n)) (X_{(n+1)j} - \mu_j(n)) \tag{7}$$

This formula gives S in a form which can be calculated independantly of the update calculation of the mean, and hence can be carried out in parallel with the mean calculation. In the network described herein the formula for unbiased covariance [5] is used:

$$\Sigma_{ij} = \frac{1}{n} S_{ij} \tag{8}$$

after n+1 points.

Formulas (5) and (7) give an iterative method for determining the mean and covariance for the case of an estimate consisting of a single Gaussian. These equations form the basis for the learning rule which we refer to as the moving mean, moving covariance formula.

On receiving an input point, each node must determine whether the point is close enough to its mean to warrant modifying its mean and covariance. This decision is based on the Mahalonobis distance (4). If the point is within some constant distance u of the mean, the node is updated using formulas (5) and (7). If no nodes report that the point is within a second constant c of their means, a new node is created, taking the input vector as its mean and a fixed starting covariance.

An important feature of the learning algorithm proposed is its one-pass nature. There is no need to present a particular input stimulus to the network repeatedly. The learning consists of updating the weights (means) and the nodal transfer functions (covariances), and the system adapts to the newest input data at the same time the information flows through the network. There may be times when it would be beneficial to have a static network, one that will not adapt to incoming data after an initial learning period. The adaptive Gaussian network can accommodate this need. However, it is in the arena of truely adaptive systems that the Gaussian network shines. Here, real-time learning is a reality.

IV. Sensor fusion / activity recognition

Evaluation of the performance of any temporal pattern recognition system is far from a straightforward task. In many, if not most, instances the incoming data does not fit exactly with any of the learned sequences, hence there is no value in a yes/no decision on the presence of a sequence. The system is asked instead to give a "best guess" of which pattern(s) it is seeing based on some set of criteria. This is by definition an ambiguous task, and the ultimate result can only be evaluated by looking at the criteria and the input data and attempting to generate a "better" solution, as defined by the human evaluator.

The network has been tested on simulated data under a variety of environmental conditions. It is necessary to recognized sequences of sensor firings as one of a set of learned sequences even if the incoming unknown sequence contains extraneous firings or missing elements (noise conditions), or multiple sequences occur simultaneously.

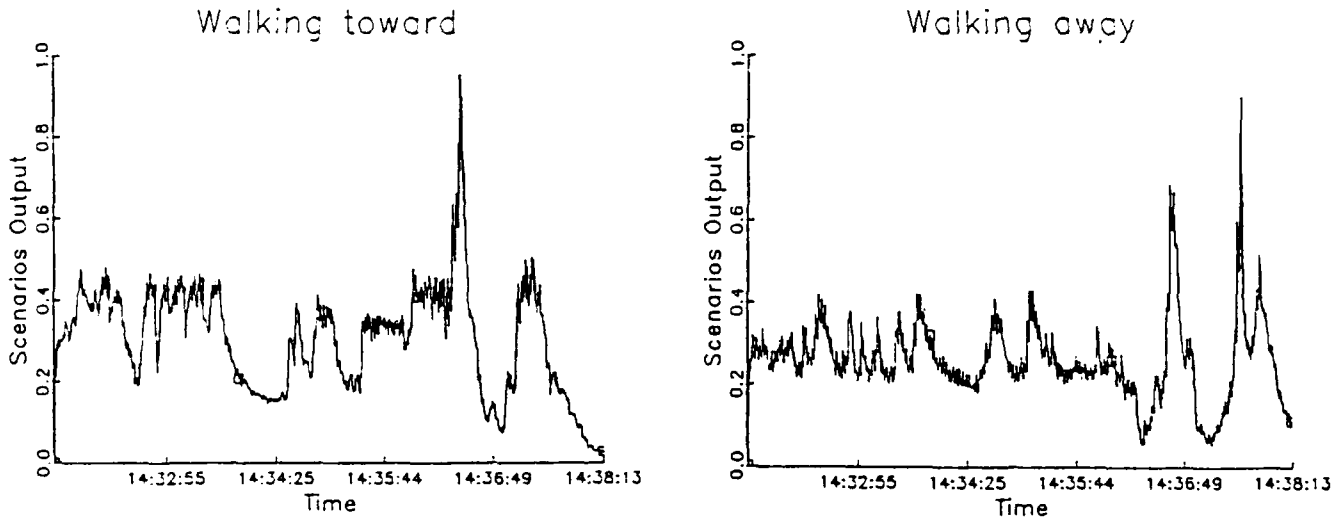
The simplest and most straightforward experiment that can be used to test the quality of the system is that of sequence recognition in a noiseless environment. Here it can be determined whether the system has performed properly or improperly. As a complete sequence is input it exactly matches one of the learned patterns, and it is imperative that the system correctly identify this pattern. In addition, the system must exhibit the ability to indicate the presence of multiple patterns. Finally, the system must be capable of prediction. Prediction is the capability to indicate the presence of a pattern prior to receiving the entire pattern. If an activity, called "burglar",

contains the exit of the burglar as part of the sequence, it would be useful if the robot indicated the activity "burglar" before the miscreant has made his escape. A system with little or no prediction capability is of limited use, since the system user would like to be warned of possible happenings prior to their conclusion, thus allowing the user to affect the ultimate outcome by acting, rather than reacting, to stimuli. The prediction capabilities are relevant in both noisy and noiseless environments.

Missing data may be caused by sensor failures or by subtle variations in the actual pattern itself which alter the sequence in some small way, yet leave the overall meaning of the pattern unaltered. Although the pattern received does not correlate exactly with the learned pattern, the network must indicate that the input stimuli is similar to one of the learned patterns. There is no hard and fast rule for determining how certain the system should be that it is seeing a given pattern. The only definitive statement that can be made is that the system must give some indication that it is close to seeing the learned pattern. The adaptive Gaussian system actually outputs a "distance" from the learned Gaussian mean to the received stimuli, using this as its analog output. Therefore the system indicates the presence of a particular pattern despite the fact that the input stimuli consists of only a partial pattern.

The system is very flexible in its use of temporal information. It can be designed to perform at various stages along a continuum from order being all-important to disregarding order. The extent to which the system can identify patterns despite the stimuli being received out-of-order rests on a design decision tightly tied to the type of environment in which the system operates. Since there exists no generic criteria for determining how important ordering should be, the network needs to be flexible in this regard.

We now look at the performance of the system on actual data. The network was taught to recognize two seperate scenarios: someone entering the room and someone exiting the room. The similarity in these scenarios figured in their selection for testing purposes. In a general sense, one is just the temporal inverse of the other. The sensors that fire when a person enters the room should be, for the most part, identical to the sensor that fire upon exiting. Roughly speaking, the major difference is that particular sensors fire in the reverse order. As such, these scenarios force the system to actually incorporate the ordering of the sensor firings, as well as which sensor fired, in order to discriminate between the two.



(a) (b)

Figure 4

Output of robot activity recognition system. (a) shows the output of the node corresponding to the event "person walking toward robot" and (b) shows the output of the node corresponding to the event "person walking away from the robot". During the time shown a person entered, approached the robot, then retreated and exited. Spurious sensor firings, inherent in a complex environment, are the cause of the high noise level seen.

The network is taught via supervised learning, using multiple instances of a scenario. Four seperate people were observed entering the room, and the sensor firings that occurred during these activities were fed to the

net along with feedback that the activity "entering" was occurring. The four "entering" scenarios are then coalesced, using the Gaussian learning scheme described above, into a template for the activity "entering". Similar teaching sequences were used for the "exiting" scenario.

At this point the network has developed a representation for both activities, and can be tested. Sensor firing data from the robot is collected over a long period of time, and this data is fed into the network. It is known where in the data certain activities occur, but the network is not given this information. The activation values for the network's output nodes are traced, and the network does in fact indicate the presence of the activities at the proper points in time (Figure 4). Notice the fact that when the network indicates the presence of "entering", it also sees "exiting", albeit with a lower likelihood. This is expected, and arises from the inverse-time relationship of the two scenarios.

V. Discussion

There are several issues that still need to be addressed. One would be to add sensors which are not strictly binary. For instance, sonar is used to aid in navigation, and it could be used also in the activity recognition. The robot has a camera mounted on top which can be used to alert the guards in the event of intruders. This camera could also provide input into the classification system.

A richer set of scenarios needs to be developed and tested on the system. There are many environmental events such as lightning, thunder, heating systems, fans, etc., which will set off some of the sensors. The system needs to be tested under these types of conditions.

Overall, the adaptive Gaussian classification system has shown itself to be a good match to the problem of multi-sensor fusion. It incorporates the ability to handle missing and noisy data with the use of temporal data to allow the system to learn a wide range of pattern classification tasks. The ability to learn sequences is essential to sensor fusion, and the system has shown itself to be quite capable in this area. Finally, its one pass learning rule and distributed processing paradigm allows real-time learning and response, making the architecture a truly dynamic system.

VI. References

[1] Chan, Golub, & LeVeque, "Algorithms for computing the sample variance: analysis and recommendations", The American Statistician, Vol. 37, No. 3, pp. 242-247, 1983.

[2] H. Everett, G. Gilbreath, S. Alderson, D. J. Marchette & C. E. Priebe, "Intelligent Security Assessment for a Mobile Sentry Robot", Proc. Institute of Nuclear Materials Management, 1988.

[3] S. Grossberg in *Pattern Recognition by Humans and Machines*, Vol 1, eds. Schwab & Nusbaum, Academic Press, 216-226, 1986.

[4] C. E. Priebe, "Temporal Information Processing", Master's Thesis, San Diego State University, 1988.

[5] G. A. F. Seber, *Multivariate Observations*, Wiley, 1984.

[6] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hill, 1986.